VampirServer 2.2

User Manual

Support / Feedback / Bugreports

Please provide us feedback! We are very interested to hear what people like, dislike, or what features they are interested in.

If you experience problems or have suggestions about this application or manual, please contact `service@vampir.eu`.

When reporting a bug, please include as much detail as possible, in order to reproduce it. Please send the version number of your copy of *VampirServer* along with the bugreport.

Please visit `http://vampir.eu` for updates and new versions.

`service@vampir.eu`
`http://vampir.eu`

Contents

1 Introduction

Vampir is a software tool for analyzing the run-time behavior of parallel software programs. It visualizes the program execution by means of event traces, gathered by monitoring software like VampirTrace, TAU, or KOJAK. The visualization takes place after the completion of the monitored program, by using data that has been captured during the program execution and stored in so-called trace files.

VampirServer is based on parallelized analysis algorithms. Data analysis and visualization are implemented as a client-server framework. VampirServer can be installed on a segment of a parallel production environment. The corresponding Vampir clients visualize the performance results graphically on remote desktop computers. Major advantages of this parallel and distributed approach are:

1. Performance data which tends to be bulky is kept at the location where it was created.
2. Parallel data processing significantly increases the scalability of the analysis process.
3. The applied performance analysis paradigm is easy to handle and works efficiently from arbitrary remote end-user platforms.
4. Very large trace files can be browsed and visualized interactively.

Vampir translates a program's performance data into a variety of graphical representations providing developers with a good understanding of performance issues concerning their parallel and serial applications. Vampir enables quick focusing on appropriate levels of detail which facilitates the detection and explanation of various performance bottlenecks such as load imbalances and communication deficiencies.

2 Getting Started

2.1 Installation and Testing

The platform specific executables for server and client, as well as their corresponding license files, are needed to run the VampirServer tool.

VampirServer is available in a platform specific installation package. Please, contact `sales@vampir.eu` for purchasing permanent licenses. Once you obtained the installation files proceed with the following steps:

1. Uncompress the gzipped tar archive
   "vampirserver-<version>-<platform>.tar.gz"
   into an installation directory of your choice by issuing the command:

   ```
   $ gtar -xzf vampirserver-<version>-<platform>.tar.gz
   ```

   or

   ```
   $ gzip -dc vampirserver-<version>-<platform>.tar.gz |
   tar -xf -
   ```

   on the command line. The result should be a directory called vampirserver-<version> which contains the following files:

   ```
   INSTALL.txt
   aux/
       vngd-start.sh
   bin/
       vngd
       vngd-config.sh
       vngd-shutdown
   doc/
       user-manual.pdf
   lib/driver/
       config.h
       MpiModCore.c
       MpiModDef.h
       MpiModProto.h
   ```

2. Adjust the system environment variables "PATH" and "LD_LIBRARY_PATH" as follows:

```
$ export PATH=$PATH:<install-dir>/bin
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:
<install-dir>/lib
```

Make sure that these variables are set accordingly whenever you want to use VampirServer. This can be achieved by adding above lines to your login scripts or by creating an appropriate “software module” for the software. Contact your local system administrator for further details.

3. The VampirServer daemon program needs to be configured for the message passing library (MPI) installed on your computer system prior to being started for the first time. You can skip this step if an MPI implementation is not available on your system or if you do not want to use it.

   A configuration script guides you through the necessary steps. You will be asked a few questions. Default answers to these questions are provided in brackets. Please confirm the default answers by simply pressing the “enter” key on your keyboard. You can alternatively enter the appropriate settings for your system. Contact your system’s administrator if you are not sure about the right answers. Enter the following commands to start the configuration:

   ```
   $ vngd-config.sh
   ```

   After successful completion, a file called “default.vngdrv” should exist in the <install-dir>/lib directory. In case of an configuration error, detailed error messages can be found in <install-dir>/vngd-config.log.

4. Copy the license file (lic.dat) you received before to <install-dir>/bin and make sure that it is readable (i.e. the right permissions bits are set) by all users.

5. Start the VampirServer daemon. VampirServer currently supports two different modes for parallel data processing. The traditional mode is based on the message passing interface (MPI) standard. It works best on machines with distributed global memory. Alternatively, VampirServer 2.0 introduces a thread-based data processing mode for shared memory based systems. While the same executable supports both modes, its invocation differs.

   a) MPI: The invocation command for the server is slightly platform dependent due to the fact that VampirServer is a program that depends on your local MPI installation. Please, consult you local MPI documentation for details. It should, in general, resemble this line:

      ```
      $ mpiexec -np <#MPI processes> vngd
      ```

      or

      ```
      $ mpirun -np <#MPI processes> vngd
      ```

This command needs to be prefixed by the appropriate command for batch job submission on systems with a batch system installed. Once the daemon is successfully running it should print out:

```
Found license file:  <install-dir>/bin/lic.dat
Running <#MPI processes -1> analysis processes...
Server listens on:  <your host name>:30000
```

If you encounter messages like:

```
Error:  Could not load communication driver
<default.vngdrv>.
```

you should check the correct setting of the environment variable "LD_LIBRARY_PATH" as described above.

If the problem still occurs, try to use the alternative environment variable "VNG_MPI_DRIVER" as follows:

```
$ export VNG_MPI_DRIVER="<install-dir>/lib/
default.vngdrv"
$ mpiexec -np <#MPI processes> vngd
```

Please note that some MPI implementations require special command line options, e. g. -x VNG_MPI_DRIVER, to propagate environment variables correctly among their processes. Consult the documentation of your MPI installation for further details.

b) Multi-threading: First of all, you need to set the environment variable "VNG_PAR_MODE" as follows:

```
$ export VNG_PAR_MODE="thread-mode"
```

Start the server program with the simple command:

```
$ vngd
```

This should result in the following message:

```
Estimating the number of processing elements
(overwrite with -n option)...
Running <#threads> analysis processes...
Server listens on:  <your host name>:30000
```

6. Start an already installed Vampir client program and establish a connection to the VampirServer daemon from within the client program by choosing "File→Remote Open...' or "File→Connect Server...".

   Complete the "Server:" and "Port:" input fields. If both daemon and client are executed on the same machine you would typically have to enter "localhost" and "30000". The daemon's startup output (see above), which tells exactly where it runs and to which port it listens, can assist you in finding the right connection parameters.

A successful connection setup is indicated in the Vampir GUI and quoted by a message “Connecting client: <hostname>:<port>” by the daemon program. Once the client program is connected to a server, its usage is identical to the stand-alone Vampir GUI. Please, consult the Vampir user manual for further reading.

7. Do further customization (optional). The process of launching VampirServer’s daemon program is slightly platform dependent. The necessary parameters and environment variables can either be set manually or automatically by a customized launch script. The latter simplifies the launch process of the daemon program to unexperienced users. You can find a generic example script under <install-dir>/aux/vngd-start.sh. Copy it to <install-dir>/bin and customize it to your needs. The script allows to specify MPI and batch system specific parameters. This procedure addresses system administrators or experienced users.

2.2 Generation of Trace Data

The generation of trace files for the VampirServer performance visualization tool requires a working monitoring system be attached to your parallel program. We recommend our *VampirTrace* monitoring facility which is available as Open Source software. More information on the installation and usage of VampirTrace can be found in the VampirTrace user guide which is available via: `http://www.tu-dresden.de/zih/vampirtrace`.

You can use the monitoring components of either TAU or KOJAK alternatively which are both free software and available for many platforms. Both monitors are able to write program traces that can be analyzed and displayed by VampirServer. They are available at: `http://www.cs.uoregon.edu/research/tau/home.php` and `http://www.fz-juelich.de/jsc/kojak`.

2.3 Starting the Server Program

Since the server program is a parallel program its invocation depends on the target platform. VampirServer currently supports two modes of parallel operation: MPI mode and thread mode. For the former, MPI has to be installed and configured properly before starting the server. The exact command line is MPI implementation-dependent. If *lam* is used, *lamboot* must be executed at the command line. If MPICH is used, a machine file might be needed (see MPICH user manual).

A server is started at the remote system that contains the trace files with the command line syntax:

```
$ mpirun -np <number of processes> vngd
```
or
```
$ mpiexec -np <number of processes> vngd
```
Please note that the number of MPI processes must be at least two. Its upper boundary is the number of traced processes.

VampirServer's new thread mode needs to be invoked **without** *mpirun* or a similar prefix. Simply type:
```
$ export VNG_PAR_MODE="thread-mode"
$ vngd
```
VampirServer will automatically detect the optimal number of threads for your system. Alternatively, the number of threads can be set manually with the "-n" command line option.

A summary of all supported command line options of the server is given in the following table:

-c	--chost=NAME	Cluster node that is going to listen for requests
-h	--help	Show this help
-n	--nthreads	Number of analysis threads (1–16) if VNG_PAR_MODE is set to "thread-mode"
-p	--port=NUMBER:[END]	Port range, the server is going to listen for requests
-v	--version	Show program version

Table 2.1: Command line options of the vngd server program

The server program will run until it is terminated either manually (with the key sequence `ctrl-c` on the command line) or automatically by your batch system. Alternatively, we provide a small utility program that triggers an internal shutdown of the server program. On systems without automatic MPI cleanup this utility can help to ensure that no orphaned processes remain on the system. Type
```
$ vngd-shutdown -p <host name>:<port>
```
on the command line to trigger a server program shutdown. The server program will terminate with the following output:
```
Server shutdown triggered by client.
```

2.4 Starting a Client

The Vampir client can be run on the same machine as the server or on another machine which is only used to view the results. Follow the instructions in the Vampir manual to start the Vampir client.

2.5 Connecting to a Server

Prior to trace data visualization, the server and client programs need to be started. Furthermore, the client program needs to be connected to the server program. Once a connection has been established between these two programs, trace files can be opened and analyzed.

Choose the entry "Open Remote. . . " or "Connect Server. . . " (depending on the client version) from the "File" menu of the client. Then determine which server and which port to use. The default for the server is "localhost" and for the port "30000". The server startup output, which tells exactly where it runs and to which port it listens, can assist you in finding the right connection parameters. If the connection could not be established successfully, the client will indicate "Connection refused" or "Connection failed".

2.6 SSH Tunneling

If the user has to connect with a remote server, ssh tunneling can be used for setting up a secure connection. To establish one via ssh the user has to use the following command line syntax:

```
$ ssh -L 30000:localhost:30000 <user>@<server-name>
```

The port which is used for communication on the client side is determined with the option "-L <portnumber>" followed by a colon and the servername (here localhost so that data is transmitted through the tunnel). Then follows another colon, the portnumber for communication on server side and the user login at the server.